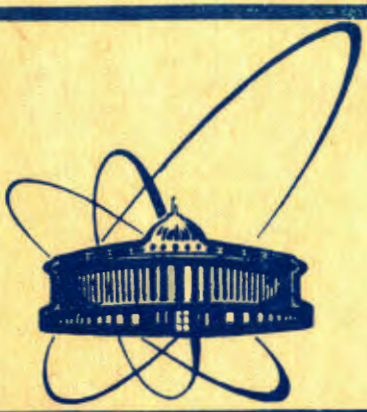

сообщения
объединенного
института
ядерных
исследований
дубна

E11-84-74

B.Naumann, M.Rudalics

TEXT PROCESSING IN A SMALL SCALE ENVIRONMENT

1984

1 Introduction

Recent developments in the standardization of graphics systems, namely the Graphical Kernel Standard (GKS) [9], and the CORE system [6], have been based on a dichotomy between modelling and viewing of graphic objects, where modelling may be considered as the specification of position, shape and size of a graphical object in the "real world", while viewing controls the appearance of the object on the display surface of a specific workstation.

A similar dichotomy (see Figure 1) exists already for some time in the text processing (or document preparation) environment with the concepts of editing and formatting, where editing confers to the specification of a document, i.e., the definition of its contents, while formatting deals with the aspect of representing the document on various output devices.

Graphics:	Modelling	→	Viewing
Text:	Editing	→	Formatting

Figure 1: Graphics vs. Text Processing Dichotomies

Modelling has been excluded from graphics standardization mainly for three reasons: (1) To maintain portability of application programs, (2) to provide device independence, i.e., the ability to support graphic output on various workstations with differing capabilities like CRT displays and plotters, and (3) by the consideration that a consensus on a viewing standard may be easier achieved than a consensus on a modelling standard. Latter has been illustrated by the fact that banning 3D-representation to the modelling part has lead to the prevalence and final sucess of GKS over the CORE system.

When seperating editing from formatting, the designers of text processing systems were lead more by pragmatic reasons, as is reflected in the development of early text formatters like RUNOFF [18], FORMAT [4], and SCRIPT [20]. Formatters had to operate in batch environments, with punched cards input and typewriter output. Only with the advent of full screen editors the technological conditions had been created to perform editing

and formatting in one step: In the emerging group of integrated editors/formatters [8], multiple dynamic windowing - ETUDE [10], PEN [3], and multiple screen techniques - JANUS [5] allow concurrent viewing of the edited and the formatted document.

Nevertheless, many structured formatters retained the factored approach. While TEX [13] or Scribe [17] perform formatting in one compound step, UNIX [12] has decomposed text processing into a number of routines, each of them taking care of one part of the formatting problem. Again a correspondence between graphics and text processing systems shows up, as is illustrated in Figure 2 with the presentation of a typical graphics pipeline and an (idealized) text processing pipeline.

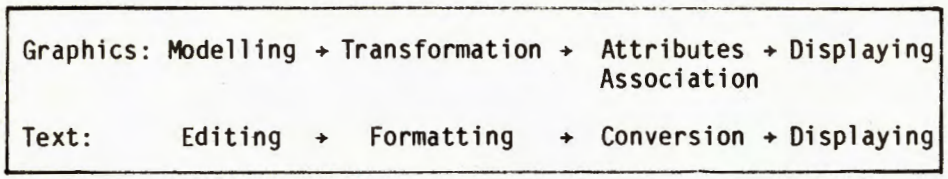

Figure 2: Graphics vs. Text Processing Pipelines

The transformation step (including normalization, segment and workstation dependent transformations) in the graphics pipeline serves for viewing purposes only; attributes association is a complex process which involves the conversion of graphics primitives according to the capabilities of the selected output device, e.g., how a specific linetype, colour or font will appear on the display surface of the workstation.

In the text processing pipeline, formatting is usually associated with the process of breaking up textual objects into lines, paragraphs and pages (including horizontal and vertical placement and alignment) while conversion essentially corresponds to the selection of a font (including lower/uppercase conversion, greek and control character substitution).

In most applications the two concluding steps in both (graphics and text processing) pipelines may be substituted by a usually implementation dependent archivation or filing step.

A realization of the idealized text processing pipeline as indicated above will, however, create problems, especially in a

small scale environment, where device dependent operations like formatting may not be performed until text conversion has been completed: It may for example be impossible, to calculate the extent of a line or paragraph before the according font has been selected. This confers to a problem in graphics pipelining, where the decision whether and how a text primitive has to be clipped, has to be postponed until its font has been determined. Therefore in the text processing system we will describe in the following sections, the formatting step from Figure 2 has been splitted into a preformatting and a postformatting step, as indicated in Figure 3.

Figure 3: Subdivision of the Text Processing Pipeline

2 Overview

The host environment of our text processing routines is an eight bit microprocessor based microcomputer development system [15], equipped with two floppy-disk drives, a low resolution video display and a lineprinter. The routines have been implemented around an existing full screen editor [7]. The main reason for using this editor was that the users of the development system (system programmers, hardware designers) were already to a high level acquainted with the editor's command language.

The basic units for manipulation by the text processing routines are: characters, words, lines, paragraphs and files.

Characters are letters, numbers, and control characters, where latter include space, return (i.e., a carriage return followed by a line feed), and form feed.

A word is any sequence of letters, numbers and printable control characters. As separators between words function space, return and form feed.

A line is a sequence of words terminated by a return. An empty line is any sequence of spaces terminated by a return.

A paragraph is a sequence of lines. Paragraphs are separated either by an empty line or by a return followed by at least one space.

A file is a sequence of paragraphs. Files are the syntactic units our routines proceed from the host operating systems's [11] point of view.

Inherently our routines provide a batch environment. Common to all routines is that they operate on a source file, usually a diskette file (but paper tape or console input - latter may be useful for testing out some situations - is also accepted) and produce a target file, which will be usually a diskette file (but may be also the lineprinter or the console). Standard extensions are defaulted for the target file, more details about invocation and naming conventions may be retrieved from [16]. Typically, a user will invoke preformatting for viewing the document on the display screen, with alternating editing and preformatting steps, perform conversion for intermediate output on the lineprinter, and use postformatting to produce the final document.

3 Preformatting

No special formatting language has been used for the formatter, therefore only implicit markup (based on spaces and returns) is accepted. Assuming that a new line is a line which starts with a space and/or is a line preceded by an empty line - the first line in a file is always a new line - the formatter operates according to the following principles:

(1) If a line on the source file is a new line and the line fits into specified pagewidth, the line will enter the target file unaltered.

(2) In all other cases the line may be appended to the previous line and/or broken, with the possible concatenation of spaces, returns and form feeds.

Line-breaking is done on a simple line-by-line basis. None of the sophisticated line-breaking systems (see [14], [2], [19]) has been implemented so far due to the restricted floating point facilities of our host system. In the future an advanced line division algorithm will be implemented and its use recommended for the postformatting step.

Pagination in preformatting is performed by inserting form feeds and a user specified amount of returns in the target file. These control characters are concatenated when the target file is newly formatted. Presently no actions are taken to avoid widows (single lines on the top or bottom of a page) in the preformatting step. Note, that formatting a target file with the same parameters that have been used when formatting the source file will result in the identic file. One parameter allows to turn on justification to the right margin (including filling with spaces). Unpaddable blanks (also referred to as significant or quoted spaces) should be designated by a character not used anywhere else in the file - usually a control character. Their final conversion will be taken care of by the postformatting step.

The reasons why we did not use an explicit markup language were (1) the user interface, as adding command lines to a file would have considerably decreased the informational content to be displayed at one time at the screen, (2) diskette space, as the present solution requires only one file for modelling and viewing purposes, and (3) the advantage that preformatted text may be again modified by the editor.

4 Conversion

Text conversion is one of the more advanced features in our system. Due to the restricted input facilities - latin uppercase only - we had to provide a number of routines which convert a file to:

(1) Latin upper/lowercase with latin lowercase as basic font,

(2) Latin/cyrillic uppercase with latin uppercase as basic font, and

(3) Latin/cyrillic uppercase with cyrillic uppercase as basic font.

To specify conversion, the user may select up to three so-called conversion specifiers (usually printable control characters which are not or little used otherwise and are eliminated in the conversion step), namely "c", "w" and "t". A conversion routine produces a target file in one of the fonts designated as basic font above. The following rules for the conversion of individual text from a basic to the opposite font apply: A letter following

a "c", all letters in a word preceded by a "w", and all letters enclosed within two "t"'s are converted to the opposite font.

Note that although the last facility - namely to surround text by two "t"'s - is rather powerful, it is also this aspect which makes it the most dangerous one to use. As "t" functions essentially as a toggle (like RUNOFF's circumflex), one unclosed "t" will cause the whole remaining text to be inverted. This situation will occur usually when either of the "t"'s has been eliminated in an editing step, e.g., when a portion of text including the "t" has been erased.

To enter any of the three conversion specifiers unaltered into the target file, it has to be preceded by a "c" in the source file. Conversion specifiers contained in a word or text to be converted remain unaltered too.

5 Postformatting

Postformatting includes pagination, selective conversion and preparation of text for printing. Pagination operates on preformatted files only, i.e., form feeds have to be already included in the source file. The pagination routines provide consecutive page numbering, alternative numbering and page headers may be requested.

Selective conversion allows to replace any character in the source file by another character on the target file, and will be usually employed for restoring unpaddable blanks.

Text directed to the lineprinter may be formatted (1) by a forwards/backwards printing routine with arbitrary pagewidth, one page per sheet, or (2) a forwards/backwards printing routine with restricted pagewidth and pagelength, two pages per sheet. Forwards/backwards printing is achieved by inserting two reserved control characters as printer directives into the target file. Printing in both directions considerably speeds up the printing process, as standard ISIS-II copy routines provide forwards printing only. While routine (1) may be used to print arbitrary text (automatic wraparound at the end of a line is included), files to be printed by routine (2) have to be preformatted.

Some optimization has been built into the printing routines, e.g., when the following line should be printed backwards and is

twice as long as the current line, the printer is directed to its initial position and the following line is printed forwards. Unlike other routines, printing procedures assume as target file the lineprinter, although an alternate diskette file may be specified by the user.

6 Conclusion

The routines described in the previous sections have been designed over the last two years and have been used in the preparation of various reports, a diploma thesis and even a dissertation. Future developments will include the elimination of widows in the preformatting step, the implementation of an advanced line-breaking algorithm and a referencing facility for bibliographic information.

References:

Reference [1] is not cited in the text.

[1] Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation. Portland, Oregon, June 1981. In: SIGPLAN Notices (ACM) 16, 6 (June 1981), and SIGOA Newsletter (ACM) 2, 1&2 (Spring/Summer 1981).

[2] Achugbue, J.O. On the Line Breaking Problem in Text Formatting. In: [1], pp.117-122.

[3] Barach, D.R., Taenzer, D.H., and Wells, R.E. Design of the PEN Editor Display Module. In: [1], pp.130-136.

[4] Berns, G.M. Description of FORMAT, a Text-Processing Program. Comm.ACM 12, 3 (March 1969), 141-146.

[5] Chamberlin, D.D., King, J.C., Slutz, D.R., Todd, S.J.P., and Wade, B.W. JANUS: An Interactive System for Document Composition. In: [1], pp.82-91.

[6] Status Report of the Graphics Planning Commitee. In: Computer Graphics (ACM/SIGGRAPH) 13, 3 (Aug. 1979).

[7] ISIS-II Credit (CRT-Based Text Editor) User's guide. INTEL Corp. 1979.

[8] Furuta, R., Scofield, J., and Shaw, A. Document Formatting Systems: Survey, Concepts, and Issues. Computing Surveys 14, 3 (Sept. 1982), 417-472.

[9] Draft International Standard ISO/DIS 7942, Information Processing Graphical Kernel System (GKS), Functional Description, Version 7.2, NI-5.9/40-82. Nov. 1982.

[10] Hammer, M., Ilson, R., Anderson, T., Gilbert, E., Good, M., Niamir, B., Rosenstein, L., and Schoichet, S. The Implementation of ETUDE, An Integrated and Interactive Document Production System. In: [1], pp.137-146.

[11] ISIS-II User's Guide. INTEL Corp., Doc.No. 98-3068,1978.

[12] Kernighan, B.W., Lesk, M.E., and Ossanna, J.F.jr. UNIX time-sharing system: Document Preparation. Bell Syst. Tech. Journal 57, 6 (July/Aug. 1978), pp.2115-2135.

[13] Knuth, D.E. TEX and Metafont: New Directions in Typesetting. Digital Press and the American Mathematical Society, Bedford, Mass., and Providence, R.I., 1979.

[14] Knuth, D.E., and Plass, M.F. Breaking Paragraphs into Lines. Software - Practice and Experience 11, 11 (Nov. 1981), 1119-1184.

[15] Intellec 800 Microcomputer Development System - Operator's Manual. INTEL Corp., Doc.No. 98-132, 1976.

[16] Naumann, B., and Rudalics, M. Text Processing Routines User's Guide. Dubna, 1983.

[17] Reid, B.K. A High-Level Approach to Computer Document Formatting. In: Conf.Rec. 7th Annual ACM Symposium on Principles of Programming Languages. Las Vegas, Nev., Jan. 1980, pp.24-31.

[18] RUNOFF. RSX-11/IAS Special Interest Group, c/o DECUS, Maynard, Ma., 1977.

[19] Samet, H. Heuristics for the Line Division Problem in Computer Justified Text. Comm.ACM 25, 8 (Aug. 1982), 564-571.

[20] Document Composition Facility: User's Guide. Form No. SH20-9161-0, IBM Corp., White Plains, 1978.

Received by Publishing Department
on February 8,1984.

Науманн Б., Рудалич М. E11-84-74

Обработка текстов на микро-ЭВМ

Интересная параллель между недавно разработанными графическими стандартами и традиционными системами форматирования текста заключается в том, что все они могут рассматриваться скорее как системы для отображения текста, чем как моделирующие системы. В данной работе описывается комплекс процедур обработки текста, включая фортматтер без отметок и программы для преобразования латинского текста в русский, которые предназначены для использования в системах разработки микро-ЭВМ типа ИНТЕЛЛЕК.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1984

Naumann B., Rudalics M. E11-84-74

Text Processing in a Small Scale Environment

An interesting parallel between recently developed graphics standards and traditional text formatting systems is that both may be considered as viewing rather than as modelling systems. This paper describes a complex of text processing procedures, including a markup less formatter and latin/cyrillic text conversion routines, which are operative in a microcomputer development system.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1984